**LAPORAN PRAKTIKUM STRUKTUR DATA**

ArrayStack



Oleh :

FAYI AMATULLAH AZHARA

NIM 2311537001

MATA KULIAH

STRUKTUR DATA

DOSEN PENGAMPU : Dr.WAHYUDI ST, MT.

**FAKULTAS TEKNOLOGI INFORMASI**

**DEPARTEMEN INFORMATIKA**

**UNIVERSITAS ANDALAS**

1. **PENDAHULUAN**

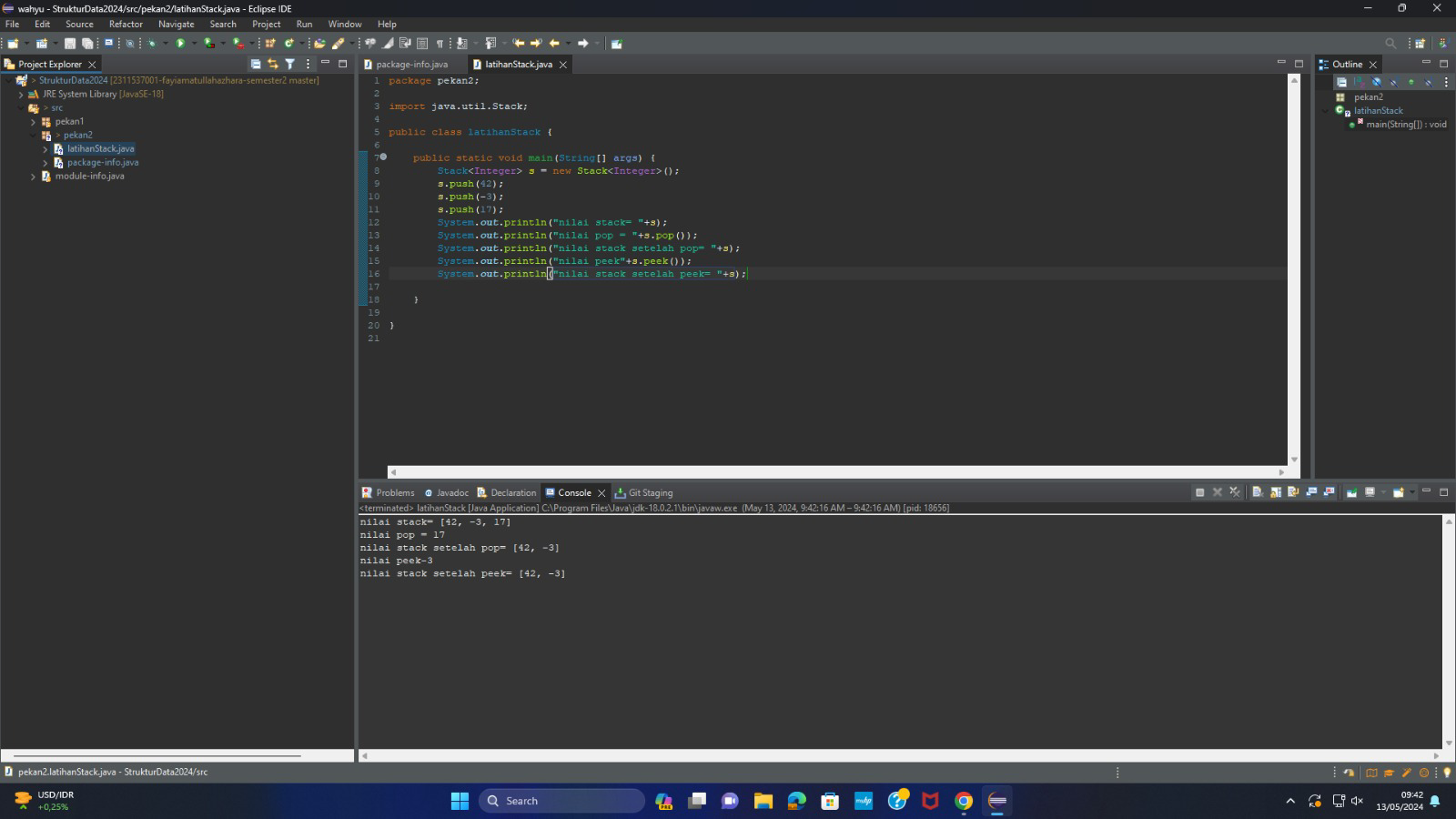
ArrayStack adalah salah satu jenis struktur data stack yang diimplementasikan menggunakan array. Stack adalah struktur data yang mengikuti prinsip LIFO (Last In, First Out), yaitu elemen yang terakhir dimasukkan ke dalam stack adalah elemen yang pertama kali akan diambil. Dalam ArrayStack, elemen-elemen disimpan dalam sebuah array dan operasi-operasi stack seperti push (menambahkan elemen), pop (menghapus elemen), dan peek (melihat elemen teratas tanpa menghapusnya) dilakukan dengan menggunakan indeks array.

1. **TUJUAN**

Mengimplementasikan penggunaan ArrayStack dalam bahasa pemrograman Java menggunakan lingkungan pengembangan Eclipse.

1. **LANGKAH – LANGKAH**
2. Membuka Eclipse IDE dan membuat proyek dengan nama “STRUKTURDATA2024”
3. Membuat Package baru dengan nama “pekan2”
4. Membuat Kelas baru dengan nama “latihanStack”

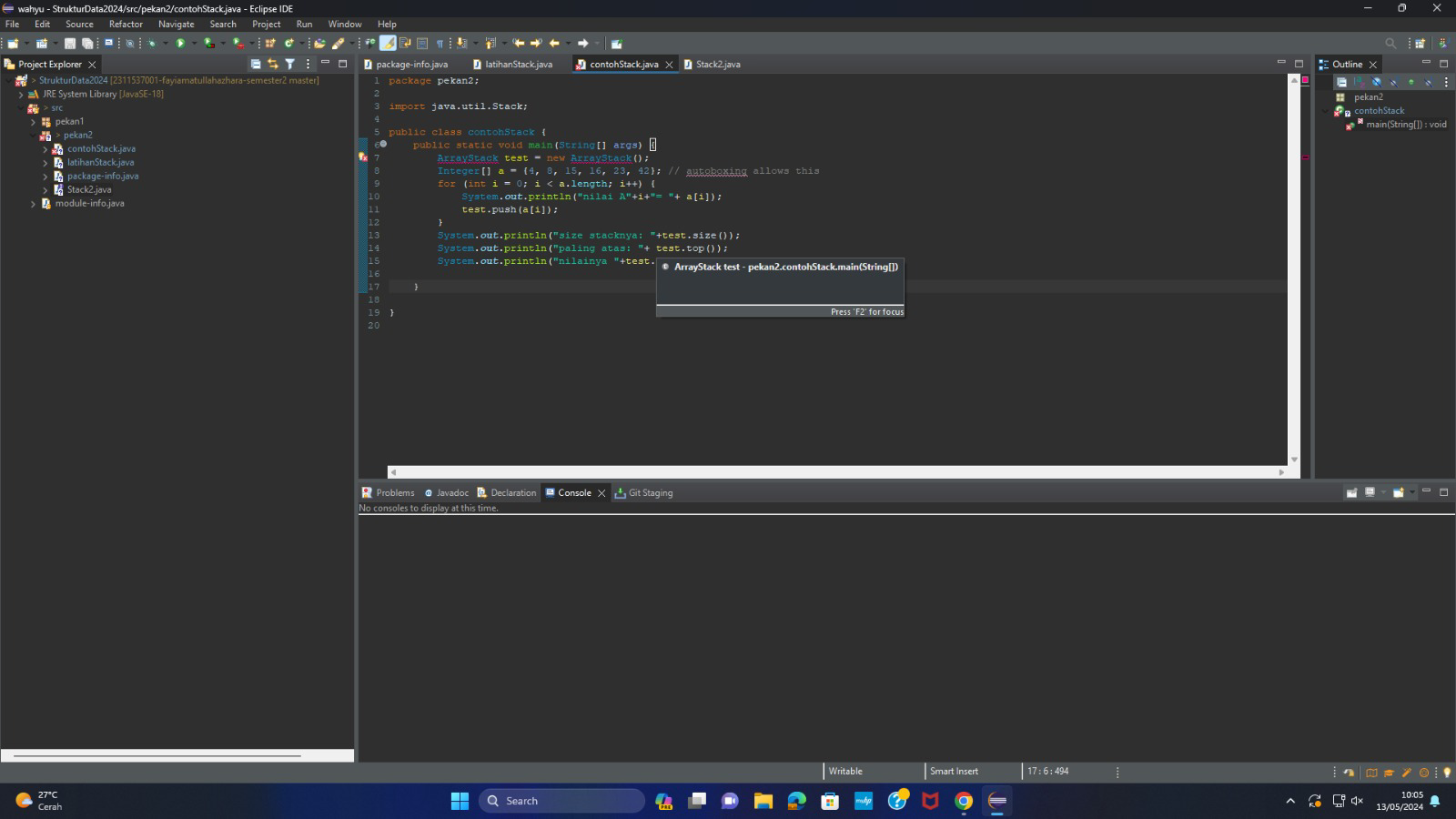
Pada kelas “latihanStack”, masukkan kode seperti gambar di bawah



Program Java di atas mendemonstrasikan penggunaan kelas Stack dari Java Collections Framework. Program ini pertama-tama membuat sebuah objek Stack yang berisi elemen bertipe Integer. Kemudian, beberapa elemen yaitu 42, -3, dan 17 dimasukkan ke dalam stack menggunakan metode push(). Setelah itu, program mencetak nilai stack saat ini. Selanjutnya, program mengeluarkan elemen teratas dari stack menggunakan metode pop(), mencetak elemen yang dikeluarkan, dan mencetak kembali nilai stack setelah elemen teratas dikeluarkan. Program kemudian menggunakan metode peek() untuk melihat elemen teratas tanpa mengeluarkannya, dan mencetak nilai stack setelah operasi peek.

1. Membuat kelas baru dengan nama “contohStack”

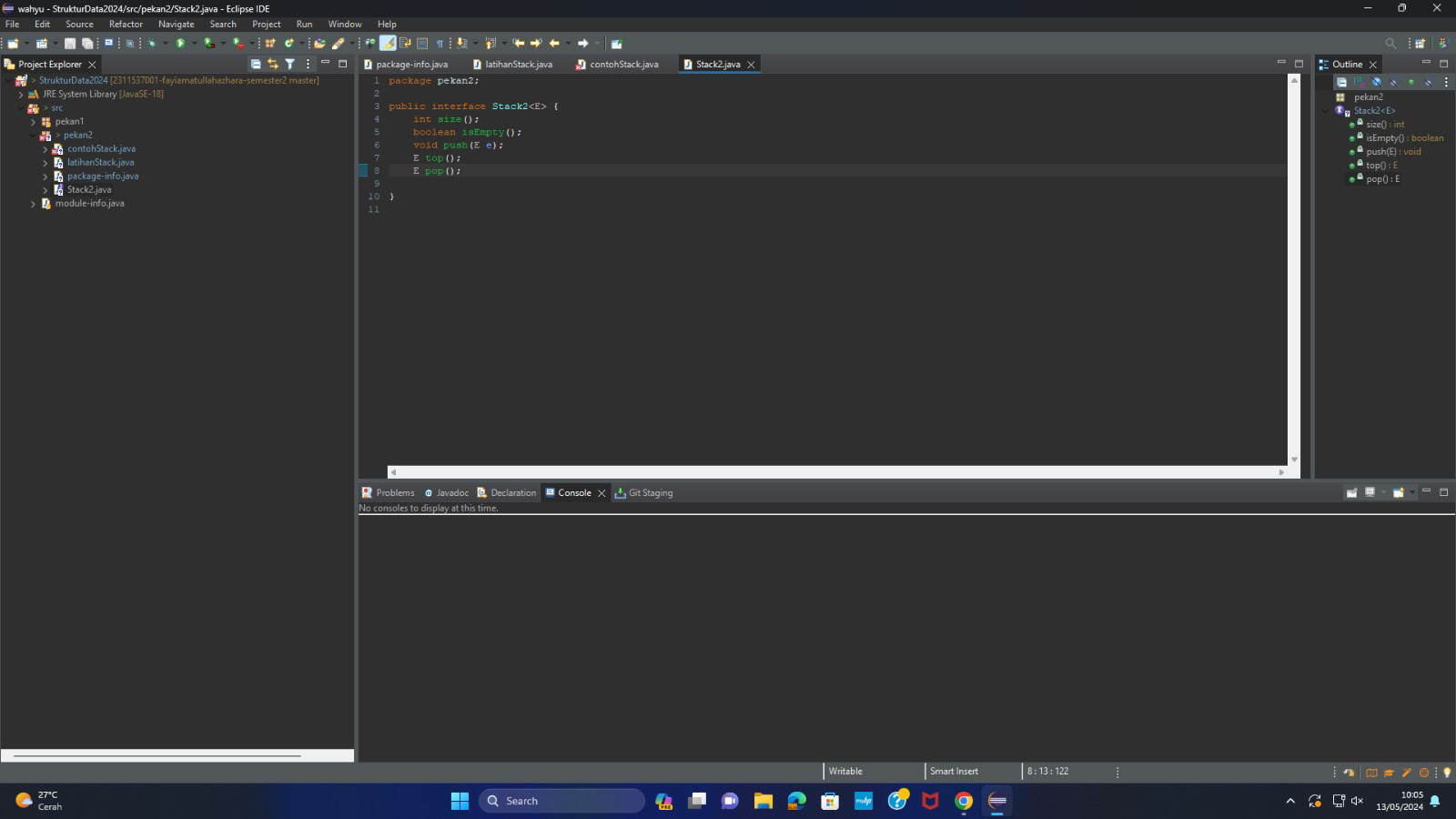
Pada kelas “contohStack”, masukkan kode seperti gambar di bawah



Program di atas akan menghasilkan error saat dijalankan karena kelas ArrayStack yang digunakan belum didefinisikan di dalam kode tersebut. Untuk memperbaikinya, kita perlu menambahkan definisi dari kelas ArrayStack.

1. Membuat kelas baru dengan nama “Stack2”

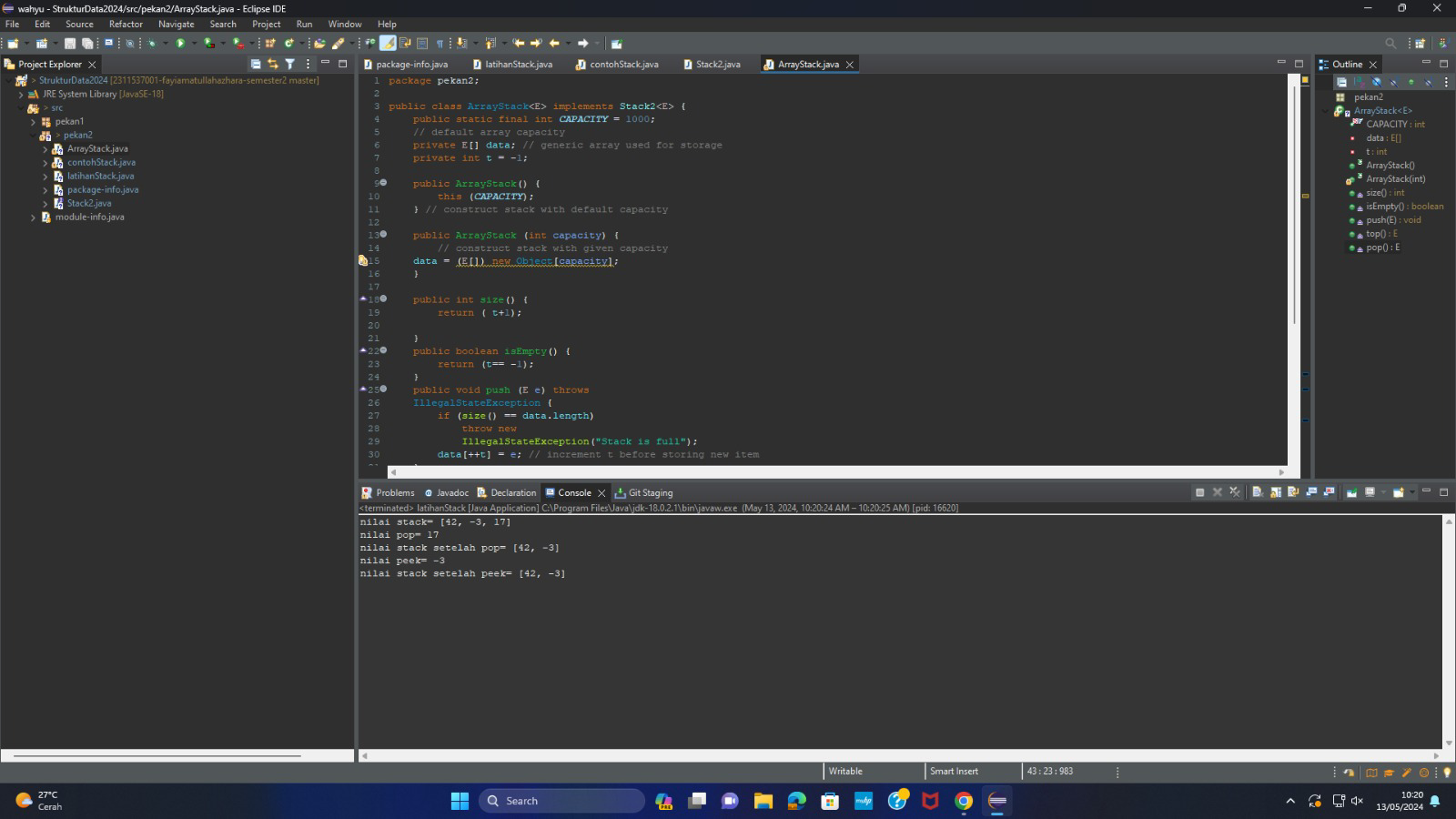
Pada kelas “Stack2”, masukkan kode seperti pada gambar di bawah

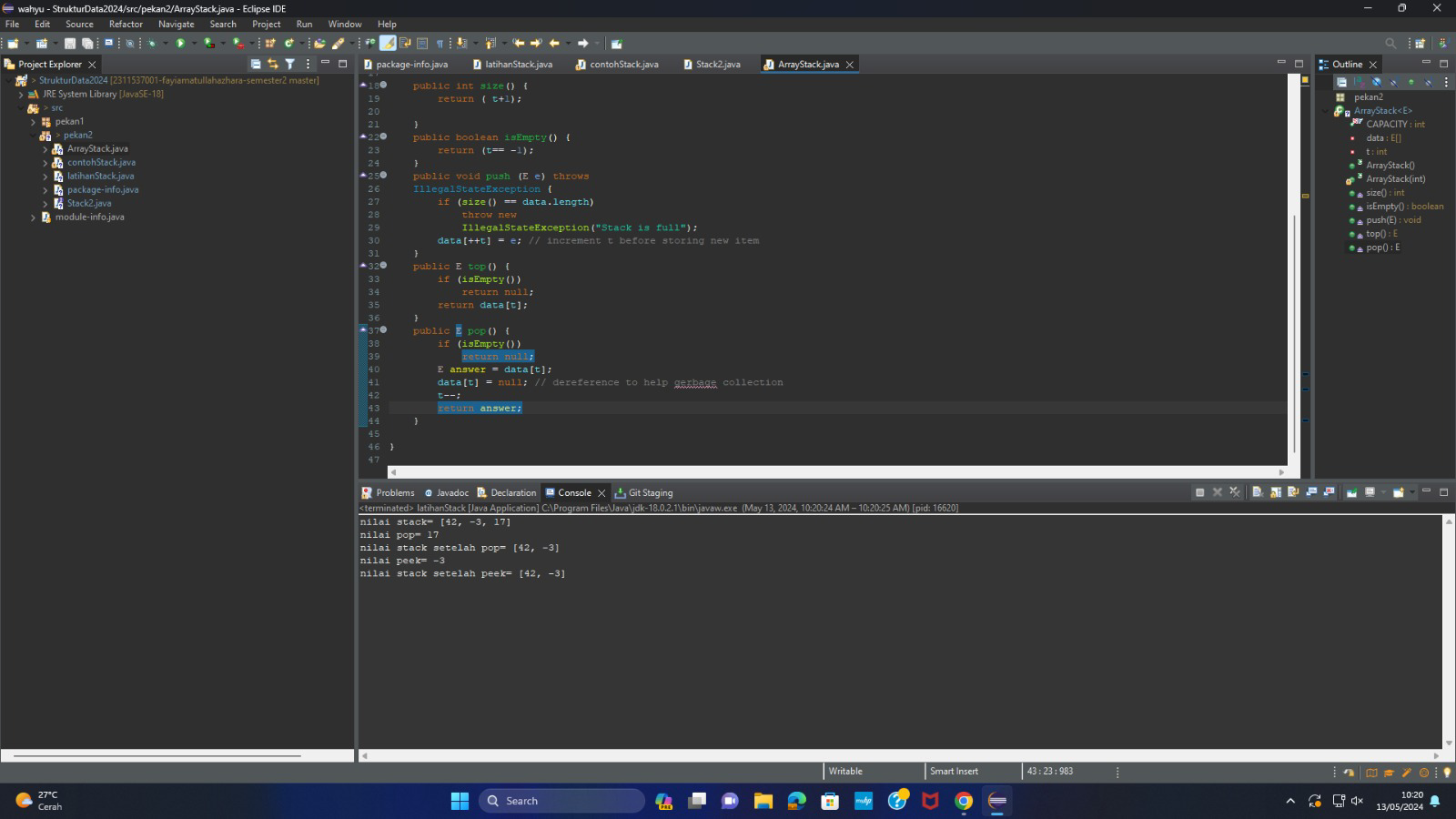


Kode ini mendefinisikan sebuah interface Stack2 dalam paket pekan2, yang mendeklarasikan metode dasar untuk struktur data tumpukan (stack) seperti size, isEmpty, push, top, dan pop. Interface ini digunakan dalam kelas contohStack, di mana sebuah objek ArrayStack dibuat dan dioperasikan dalam metode main. Kelas ArrayStack seharusnya mengimplementasikan interface Stack2 dengan menggunakan ArrayList untuk menyimpan elemen-elemen tumpukan, sehingga menyediakan implementasi konkret untuk metode-metode yang dideklarasikan dalam Stack2. Dalam contohStack, beberapa elemen integer dimasukkan ke dalam ArrayStack, dan metode size, top, serta pop digunakan untuk menampilkan ukuran tumpukan, elemen teratas, dan menghapus elemen teratas, masing-masing, guna mendemonstrasikan fungsi dasar dari tumpukan.

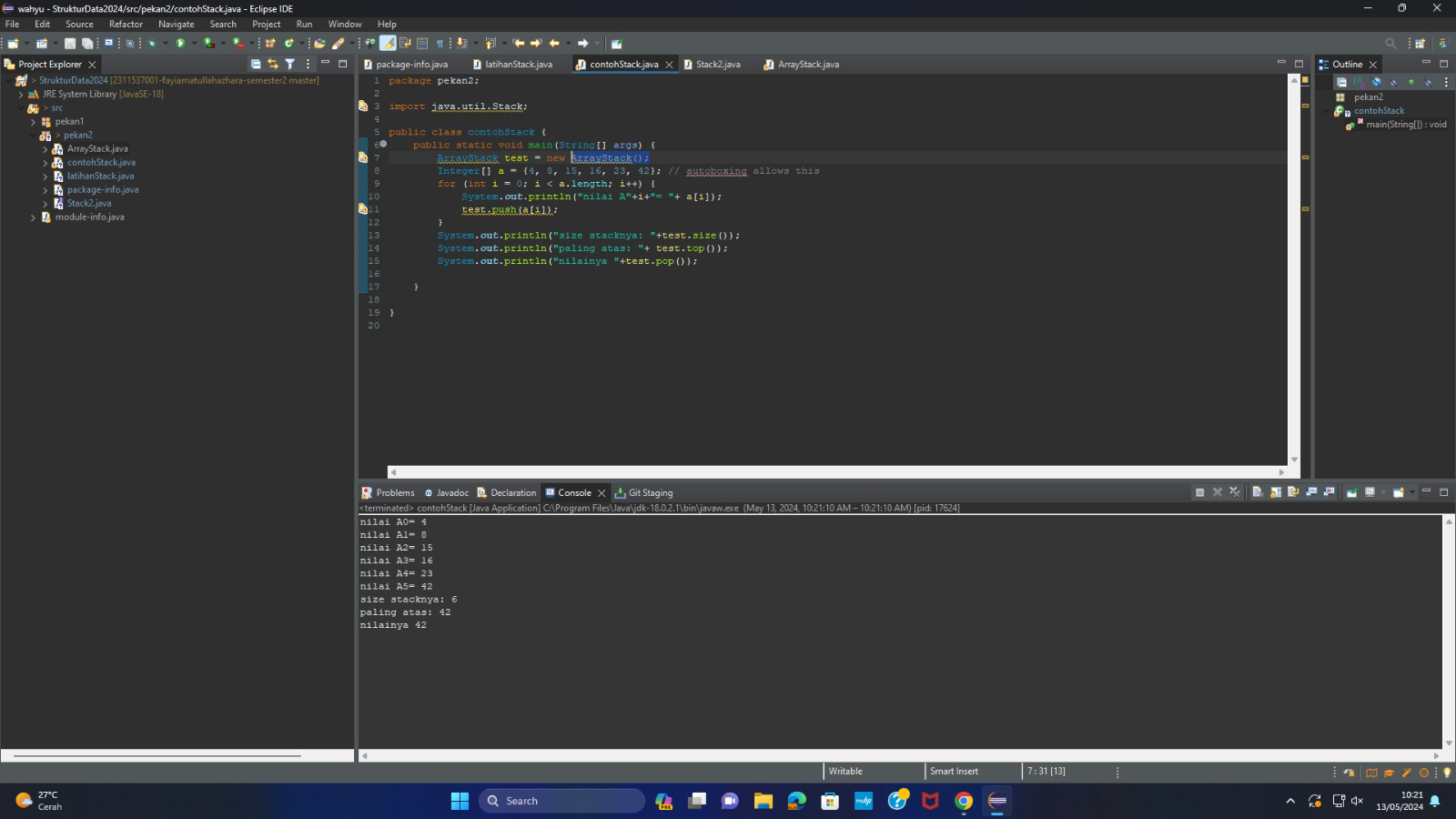
1. Membuat kelas baru dengan nama “ArrayStack”

Pada kelas “ArrayStack”, masukkan kode seperti gambar di bawah



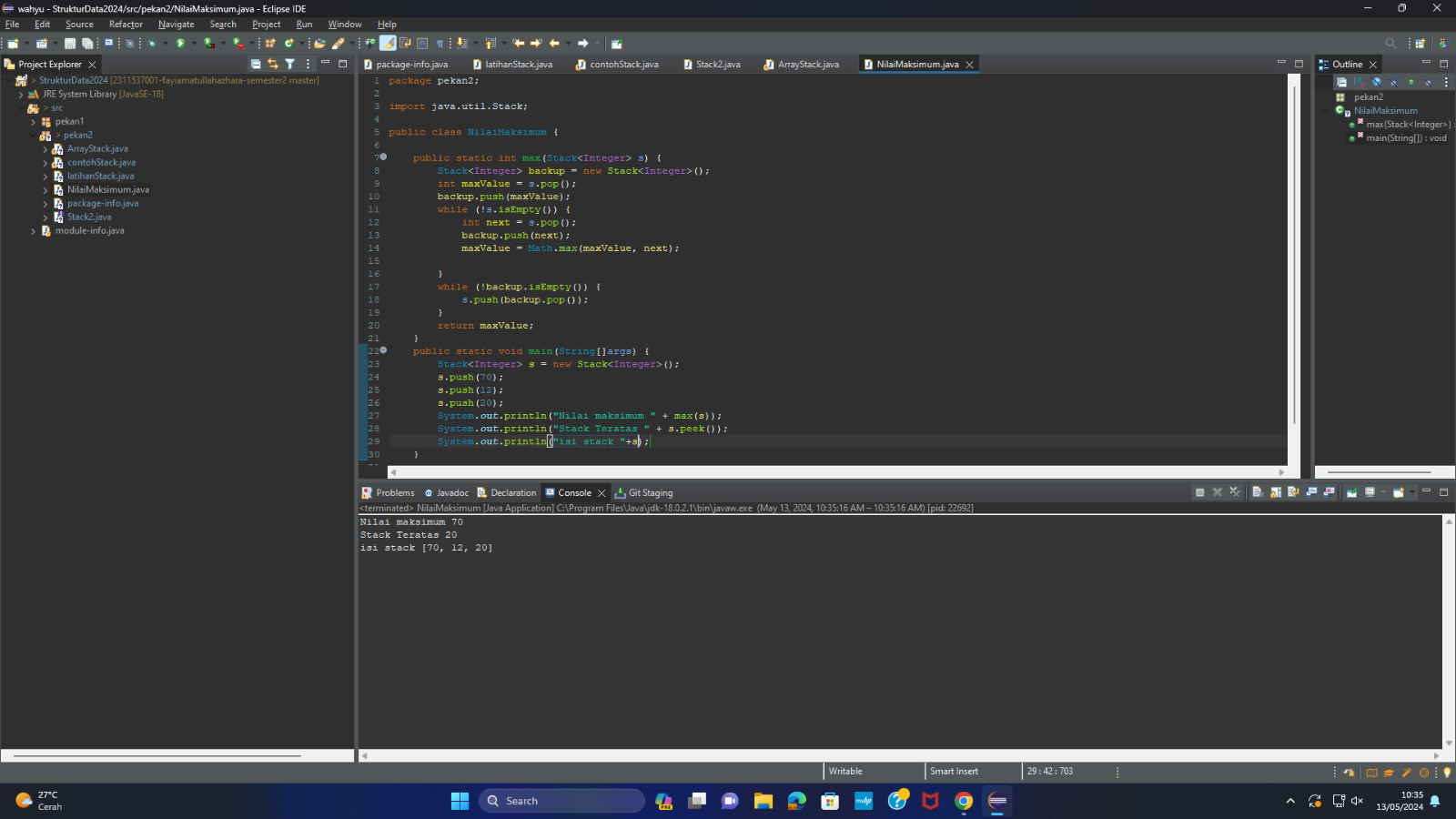


Kelas `ArrayStack` dalam paket `pekan2` adalah implementasi konkret dari interface `Stack2`, yang mengatur struktur data tumpukan (stack). Kelas ini menggunakan array generik `data` untuk menyimpan elemen-elemen tumpukan dan variabel `t` untuk melacak posisi atas tumpukan. Konstruktor kelas `ArrayStack` memungkinkan penggunaan kapasitas default atau kapasitas yang ditentukan pengguna saat membuat objek. Metode `size()` mengembalikan jumlah elemen dalam tumpukan, `isEmpty()` memeriksa apakah tumpukan kosong, `push(E e)` menambahkan elemen ke tumpukan (dengan pengecualian jika tumpukan penuh), `top()` mengembalikan elemen teratas tanpa menghapusnya, dan `pop()` menghapus dan mengembalikan elemen teratas. Metode `pop()` juga membersihkan referensi elemen yang dihapus untuk membantu pengelolaan sampah.

maka program yang tadinya error pada kelas “contohStack” akan berjalan jika di run dan menghasilkan output seperti di bawah  


1. Membuat kelas baru dengan nama “NilaiMaksimum”

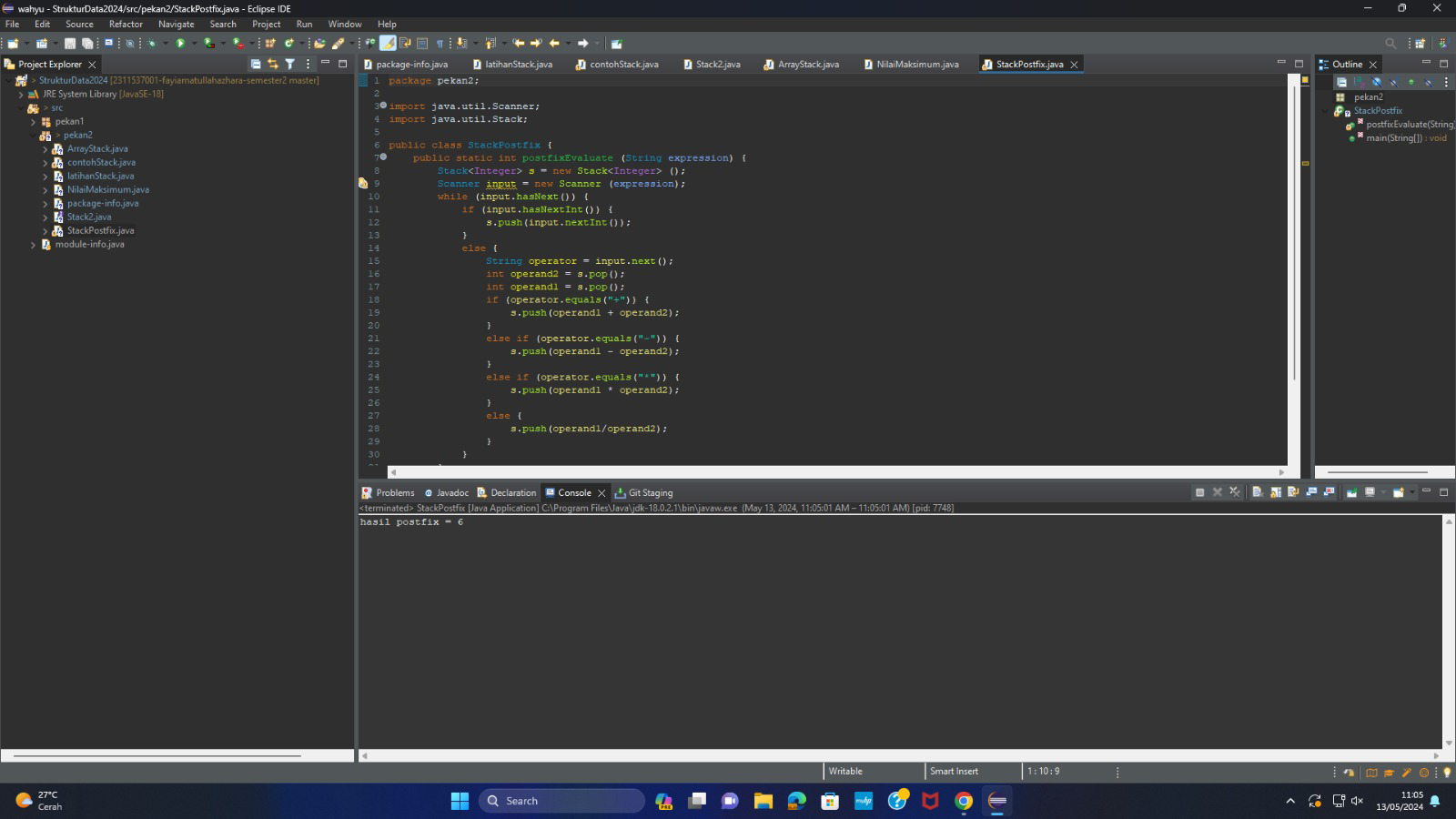
Pada kelas “NilaiMaksimum”, masukkan kode seperti pada gambar di bawah

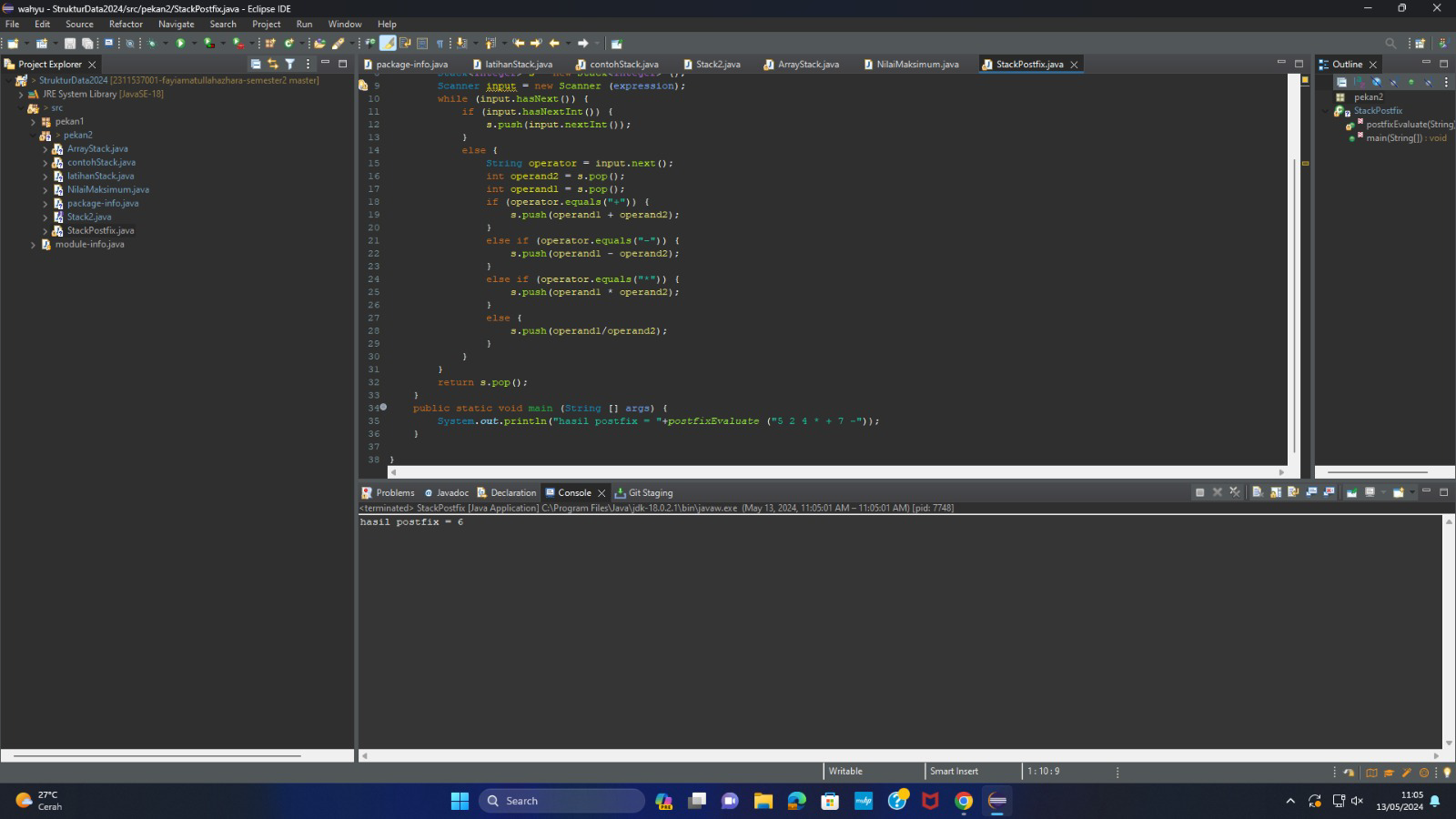


Kode dalam kelas `NilaiMaksimum` di paket `pekan2` adalah implementasi dari sebuah metode `max`, yang mengambil sebuah tumpukan (stack) yang berisi bilangan bulat dan mengembalikan nilai maksimum di dalamnya. Metode ini menggunakan dua tumpukan: `s` sebagai tumpukan awal dan `backup` sebagai tumpukan sementara untuk menyimpan nilai-nilai yang telah diambil dari `s`. Algoritma yang digunakan adalah dengan mengeluarkan elemen-elemen dari `s` satu per satu, membandingkan nilai maksimum dengan nilai selanjutnya, dan menyimpan nilai maksimum dalam variabel `maxValue`. Setelah mencari nilai maksimum, elemen-elemen yang telah diambil dari `s` dimasukkan kembali ke dalamnya dari `backup`. Dalam metode `main`, sebuah tumpukan `s` dibuat, diisi dengan beberapa nilai, kemudian nilai maksimum dari tumpukan tersebut dicetak bersama dengan nilai teratas tumpukan dan isi keseluruhan tumpukan.

1. Membuat kelas baru dengan nama “StackPostfix”

Pada kelas “StackPostfix”, masukkan kode seperti pada gambar di bawah





Kode dalam kelas `StackPostfix` di paket `pekan2` adalah implementasi dari metode `postfixEvaluate`, yang menghitung nilai dari sebuah ekspresi postfix (postfix expression) yang diberikan. Metode ini menggunakan tumpukan (`Stack<Integer> s`) untuk mengevaluasi ekspresi postfix. Algoritma yang digunakan adalah dengan membaca setiap token dari ekspresi postfix menggunakan objek `Scanner`, kemudian memeriksa apakah token tersebut adalah bilangan bulat atau operator. Jika token adalah bilangan bulat, maka bilangan tersebut dimasukkan ke dalam tumpukan. Jika token adalah operator (`+`, `-`, `\*`, `/`), maka dua operand teratas di tumpukan diambil, dihitung sesuai dengan operatornya, dan hasilnya dimasukkan kembali ke dalam tumpukan. Setelah selesai membaca seluruh token dalam ekspresi postfix, hasil evaluasi terdapat pada tumpukan dan akan diambil menggunakan `s.pop()` untuk dikembalikan sebagai hasil dari metode `postfixEvaluate`.

Dalam metode `main`, sebuah ekspresi postfix ("5 2 4 \* + 7 -") diberikan sebagai argumen kepada `postfixEvaluate`, dan hasil evaluasi dari ekspresi tersebut dicetak menggunakan `System.out.println`.

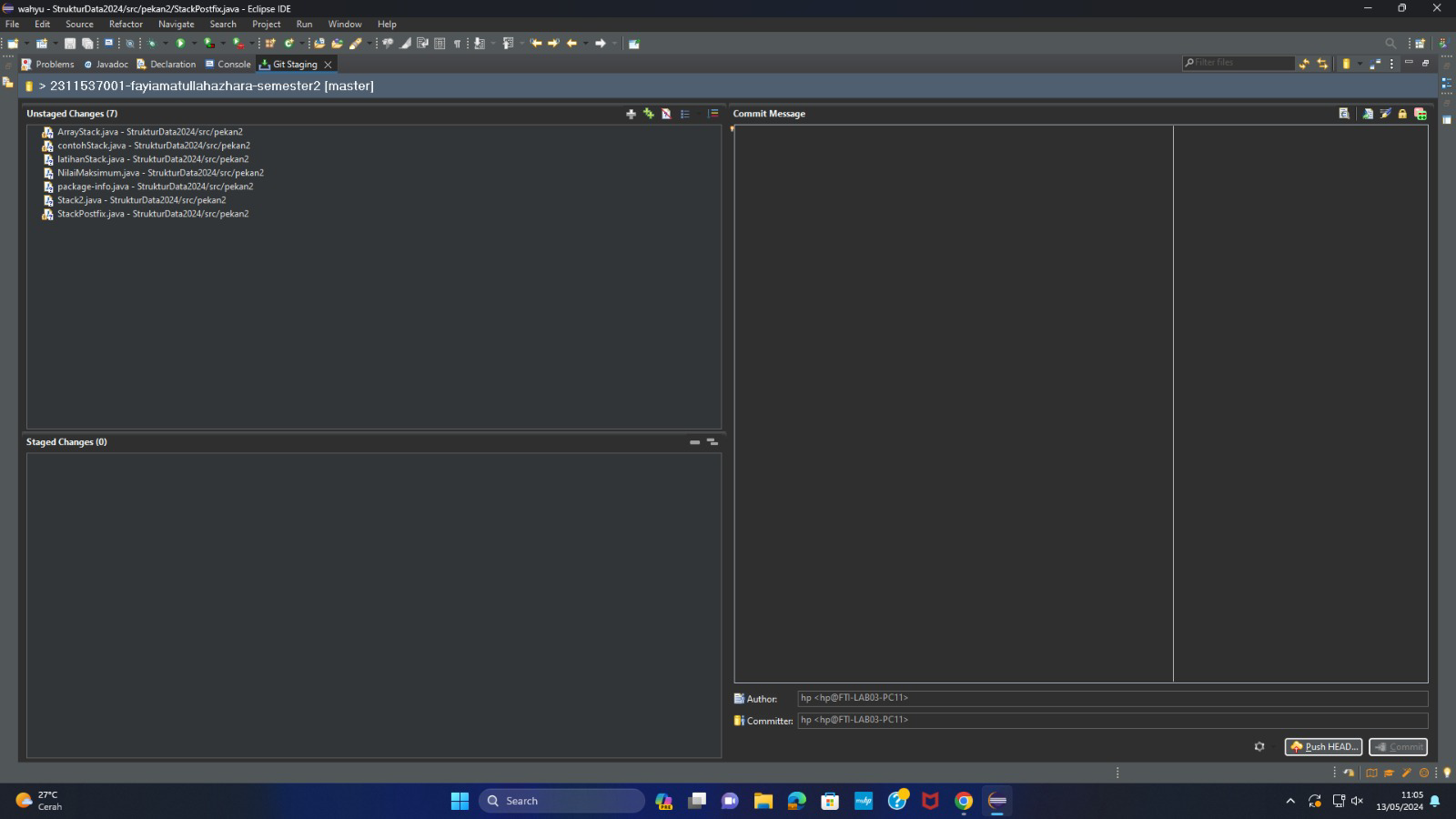
Langkah-langkah =

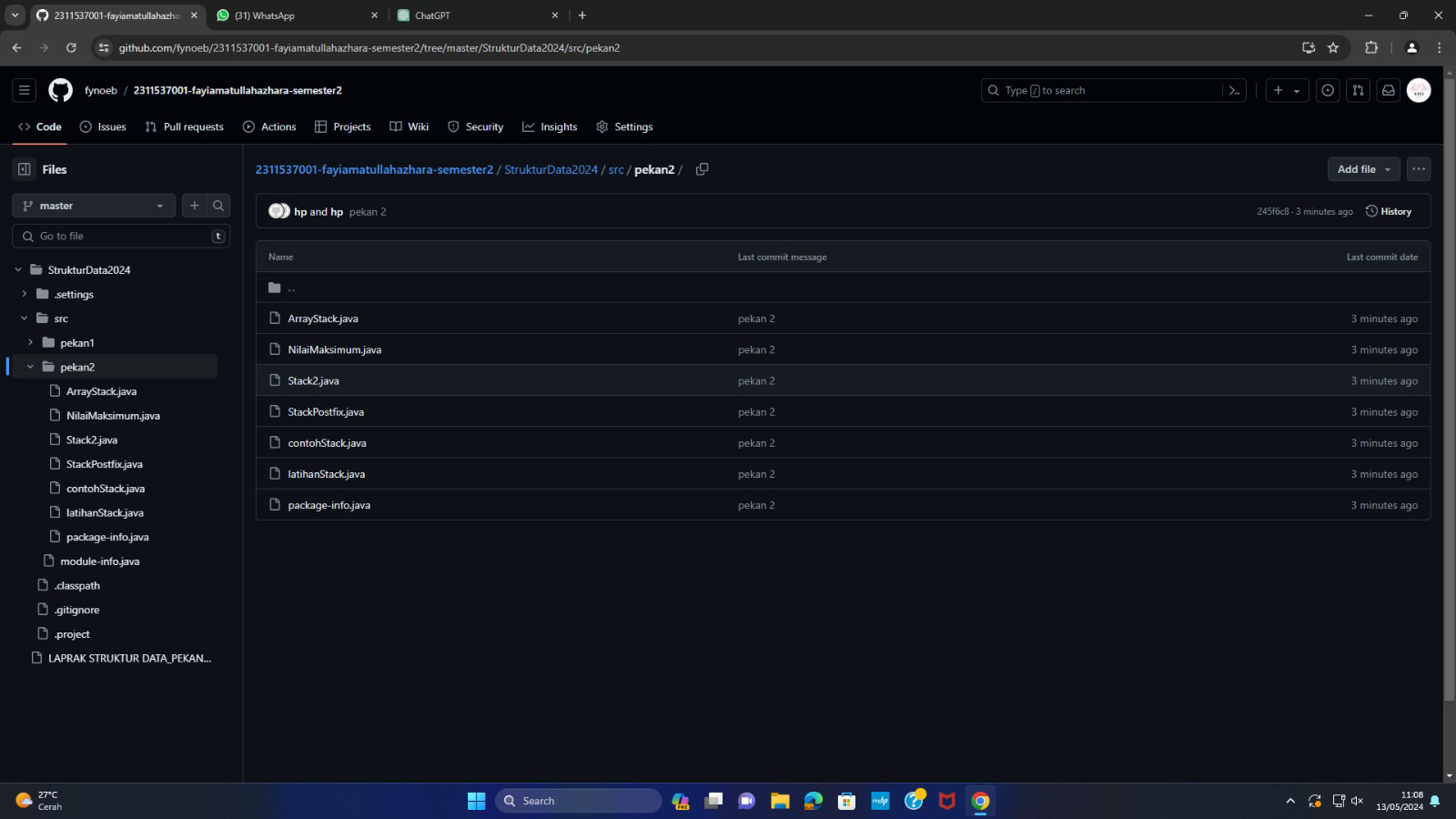
1. Push 5 ke dalam tumpukan.
2. Push 2 ke dalam tumpukan.
3. Push 4 ke dalam tumpukan.
4. Pop 4 dan 2 dari tumpukan, lalu hitung 4 \* 2 = 8, kemudian push hasilnya (8) ke dalam tumpukan.
5. Pop 8 dan 5 dari tumpukan, lalu hitung 8 + 5 = 13, kemudian push hasilnya (13) ke dalam tumpukan.
6. Push 7 ke dalam tumpukan.
7. Pop 7 dan 13 dari tumpukan, lalu hitung 13 - 7 = 6,

* kemudian push hasilnya (6) ke dalam tumpukan.

Akhirnya, hasil evaluasi dari ekspresi "5 2 4 \* + 7 -" adalah 6.

1. Mengimpor projek “STRUKTUR DATA 2024” ke github dengan repository baru “2311537001-fayiamatullahazhara-semester2”





1. **PENUTUP**

Dalam laporan praktikum ini, saya telah menjelaskan implementasi struktur data ArrayStack menggunakan bahasa pemrograman Java dalam lingkungan pengembangan Eclipse. Langkah-langkah praktikum meliputi pembuatan proyek, package, dan kelas-kelas seperti `latihanStack`, `contohStack`, `Stack2`, `ArrayStack`, `NilaiMaksimum`, dan `StackPostfix`. Pada bagian paling akhir, saya juga memperbaiki error yang terjadi pada kelas `contohStack` dengan menambahkan definisi dari kelas `ArrayStack`. Melalui implementasi ini, saya dapat memahami konsep dasar dari struktur data tumpukan (stack), penggunaan array dan generic di dalamnya, serta penggunaan tumpukan dalam evaluasi ekspresi postfix. Kesimpulan dari praktikum ini adalah pemahaman yang lebih baik tentang konsep struktur data tumpukan dan kemampuan untuk mengimplementasikan konsep tersebut dalam bahasa pemrograman Java dengan bantuan lingkungan pengembangan Eclipse.